COMSC-210 Lab 10
Hash Tables

GOOD PROGRAMMING PRACTICES *show / hide*

ABOUT THIS ASSIGNMENT
In this assignment you will write a templated ADT hash table class, and test it. The version of the hash table implementation that you will write in this lab will be based on chaining, with a static array of STL `lists`. You apply this class to the classic Game Of Life simulation.

After you complete this lab assignment, post the required files to the COMSC server so that you receive the proper credit for this 50-point lab. Your posted files are due at midnight of the evening of the due date indicated in the course outline. Use the "Submit your FA2015 work" link on the class website to post your file for this lab, using the **lab10** folder provided.

LAB 10: Write, Test, and Apply The HashTable Class Template [`HashTable.h`]
Purpose. In this lab you will implement the hash table interface outlined in the lecture topic #10 notes.

Requirements. Write **`HashTable.h`**. Follow the specification for the public interface in the online lecture notes to write the H file. Do NOT use the textbook's implementation of the hash table -- use chaining as taught in this class. You should write a test driver like you did for lab 9's AssociativeArrayDriver.cpp, but do NOT submit it.

In your test driver, use a capacity at least 25% larger than the number of values you expect to store. And you'll have to figure out how to write a hash code function for `struct TermSection`.

Once the template is tested, apply it to the classic **Game Of Life Simulation**. Right-click here to download a Game Of Life simulation that should work with your **HashTable.h**. Your lab 10 Game Of Life results should match your lab 9's exactly.

Submit one file (the template H) to the class website for credit. Do NOT submit the test driver or GameOfLife10.cpp -- your H should work with my test driver and the downloaded Game Of Life CPP.

LAB 10b: Apply The STL `map` Template To The Classic Game Of Life Simulation [`GameOfLife10b.cpp`]
Purpose. In this lab you will use the STL map template to do the Game Of Life simulation.

Requirements. Rewrite the downloaded `GameOfLife10.cpp` from above as `GameOfLife10b.cpp`, using the STL `map` template instead of your own hash table implementation. There should be no include for *your* HashTable in this version -- just the STL map. Here are the differences:

1. The STL map does not use a hashCode function, so you won't need that.
2. The STL map does not need `operator==` for its key, but it does need `operator<`. You decide how to write it.
3. The STL map does not have containsKey, so you'll have to find its equivalent in the STL documentation online.

HINT: In the operator less-than function, devise a formula that allows each row-column combination to be *unique* over their expected ranges. For example, don't do something like `row+col;` because several different combinations of row and column would result in the same number.

Program I/O. Same as lab 9's and 10's (above) Game Of Life. But note this -- do NOT just run the 2nd pattern to the 24th generation and believe that the Game Of Life for labs 9 and 10 match. Run them to many future generations to convince yourself that they do in fact yield identical results.

How to pause a console program: *show / hide*

GRADING RUBRIC *show/hide*